École d'Ingénieurs du
Littoral-Côte-d'Opale

TP : Réseaux industriels et supervision

Interface Pygame

Auteur: *Fono Colince*

Supervisé par:
Mr. Pierre Chatelain

Date:
May 26, 2024

Vous pouvez trouver le code source complet du projet sur GitHub à l'adresse suivante :
https://github.com/Fonocol/pygame/
Vous pouvez également visionner une vidéo de démonstration de la simulation de réservoir de liquide en suivant ce lien :
https://github.com/Fonocol/pygame/raw/main/Assetes/video.mp4

Contents

1 Introduction **2**

2 Interface Pygame **2**
2.1 Interface principale 2
2.2 Cycle de remplissage 3
2.3 Cycle de vidange 4

3 Annexe **5**
3.1 La liste des icônes créées 5
3.2 Un diagramme des classes 5
3.3 codes sources 6
3.3.1 sources 6
3.3.2 code python 7

1 Introduction

Dans ce TP, nous avons développé une interface de simulation de réservoir de liquide en utilisant Pygame. Cette interface simule un système de contrôle industriel et permet de visualiser le niveau de liquide dans le réservoir, ainsi que de contrôler différents composants du système.

2 Interface Pygame

2.1 Interface principale

Figure 1: Interface principale de la simulation de réservoir de liquide

Dans cette interface, l'utilisateur peut lancer une simulation en cliquant sur le bouton SART ou QUIT pour quiter la simulation

2.2 Cycle de remplissage

Figure 2: Vue pendant le cycle de remplissage du réservoir de liquide.

Cette image montre le réservoir de liquide pendant le cycle de remplissage, où le niveau de liquide augmente progressivement jusqu'à atteindre la capacité maximale du réservoir.

2.3 Cycle de vidange

Figure 3: Vue pendant le cycle de vidange du réservoir de liquide.

Dans cette image, on peut observer le réservoir de liquide pendant le cycle de vidange, où le niveau de liquide diminue progressivement jusqu'à ce que le réservoir soit complètement vidé.

3 Annexe

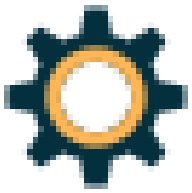
3.1 La liste des icônes créées

Dans notre interface, nous avons utilisé plusieurs icônes pour représenter différents états des composants. Voici les icônes utilisées :

- **Icône représentant une Ampoule activé:**

- **Icône représentant une Ampoule désactivé:**

- **Icône représentant une roue:**



3.2 Un diagramme des classes

Il est à noter que nous n'avons pas utilisé de programmation orientée objet (POO) dans ce projet, car la structure du code ne le nécessitait pas. Nous avons plutôt utilisé une approche procédurale pour développer l'interface de simulation.

3.3 codes sources

3.3.1 sources

Vous pouvez trouver le code source complet du projet sur GitHub à l'adresse suivante :
https://github.com/Fonocol/pygame/

Vous pouvez également visionner une vidéo de démonstration de la simulation de réservoir de liquide en suivant ce lien :
https://github.com/Fonocol/pygame/raw/main/Assetes/video.mp4

3.3.2 code python

```python
import pygame
import time
from math import pi, cos, sin, sqrt
import random
from modbus import Modbus
import sys

# initialisation de Pygame et de l'ecran
pygame.init()
modbus = Modbus()
ecran = pygame.display.set_mode((800, 650))
pygame.display.set_caption("Simulation de Reservoir de Liquide")
clock = pygame.time.Clock()
background_color = (30, 30, 30)
background_surface = pygame.Surface((800, 650))


# Couleurs
GREEN = (0, 255, 0)
BLACK = (0, 0, 0)
RED = (255, 0, 0)
BLUE = (0, 0, 255)
WHiTE = (255, 255, 255)
LiQUiDE = (random.randint(1, 254), random.randint(1, 254), 255)
GREY = (60, 50, 90)
AMPOULE = [pygame.image.load('./Assetes/Aon.png'), pygame.image.
    load('./Assetes/Aoff.png')]

# Fonctions pour les courbes
def f(x, r, x0, y0):
    return (x + x0, -(sqrt(r - x**2)) + y0)
def g(x, r, x0, y0):
    return (x + x0, -(sqrt(r**2 - (x - r)**2)) + y0)

def draw_path(surface, x0, y0, dir):
    x0 = x0 + (dir * 80)
    y0 = y0 + 70
    if dir == 1:
        pos = [(f(i * 0.01, 10000, x0, y0)[0], f(i * 0.01, 10000,
     x0, y0)[1]) for i in range(0, 10001, 100)]
    else:
        pos = [(g(i * 0.01, 100, x0, y0)[0], g(i * 0.01, 100, x0,
     y0)[1]) for i in range(0, 10001, 100)]
    for p in pos:
        pygame.draw.circle(surface, LiQUiDE, (p[0], p[1]), random
    .randint(2, 5))

# Generation initiale des bulles
def generate_bubbles(num_bubbles):
    return [(random.randint(10, 90), random.randint(10, 90),
    random.randint(2, 5)) for _ in range(num_bubbles)]

# initialisation des bulles
bubbles = generate_bubbles(15)

```

```python
# Animation des bulles
def animate_bubbles(bubbles, height):
    for i in range(len(bubbles)):
        bubbles[i] = (bubbles[i][0], (bubbles[i][1] - 0.5) % 100, bubbles[i][2])

# Fonction pour dessiner un indicateur de niveau de liquide
def draw_level_indicator(surface, x, y, fill_level):
    font = pygame.font.Font(None, 24)
    level_text = font.render(f"{fill_level:.1f}%", True, BLACK)
    surface.blit(level_text, (x, y))

# Fonction pour dessiner le reservoir
def draw_tank(surface, x, y, width, height, fill_level):
    # Dessin du liquide
    liquid_height = fill_level / 100 * height
    pygame.draw.rect(surface, LiQUiDE, (x + 2, y + height - liquid_height, width, liquid_height), 1)
    # Ajout d'un effet de degrade pour le liquide
    for i in range(1, int(liquid_height), 2):
        alpha = 255 - int(255 * (i / liquid_height))
        s = pygame.Surface((width, 2), pygame.SRCALPHA)
        s.fill((LiQUiDE[0], LiQUiDE[1], 255, alpha))
        surface.blit(s, (x + 2, y + height - i))
    # Tank
    pygame.draw.line(surface, BLACK, (x, y), (x, y + height), 3)
    pygame.draw.line(surface, BLACK, (x, y + height), (x + width, y + height), 3)
    pygame.draw.line(surface, BLACK, (x + width, y + height), (x + width, y), 3)
    # Dessiner et animer les bulles
    draw_bubbles(surface, x, y, width, height, fill_level, bubbles)
    animate_bubbles(bubbles, 10)

def draw_bateri(surface, x, y, width, height, fill_level):
    # Dessin du liquide
    col = LiQUiDE
    liquid_width = fill_level / 100 * width
    if liquid_width < width / 2:
        col = RED
    else:
        col = LiQUiDE
    pygame.draw.rect(surface, col, (x, y, liquid_width, height))
    pygame.draw.rect(surface, BLACK, (x, y, width, height), 2)
    draw_level_indicator(surface, x + width, y, fill_level)

def draw_indicator(surface, x, y, l1, l2, l3):
    draw_bateri(surface, x, y, 400, 20, l1)
    draw_bateri(surface, x, y + 30, 400, 20, l2)
    draw_bateri(surface, x, y + 60, 400, 20, l3)

# Fonction pour dessiner des cercles (bubbles) dans le liquide
def draw_bubbles(surface, x, y, width, height, fill_level, bubbles):
    liquid_height = fill_level / 100 * height
    for bubble in bubbles:
```

```
        bubble_y = y + height - bubble[1] * liquid_height / 100
        pygame.draw.circle(surface, LiQUiDE, (x + bubble[0] *
    width / 100, int(bubble_y)), bubble[2])

def rotate_image(ecran, x, y, image, angle):
    rotated_image = pygame.transform.rotate(image, angle)
    rot_img_rect = rotated_image.get_rect(center=(x, y))
    ecran.blit(rotated_image, rot_img_rect)

def tapie(ecran, x, y, angle, img1, dir, stat, ampoule):
    ampouleRect = ampoule[stat].get_rect()
    if dir == 1:
        ampouleRect.x = x - 60
    else:
        ampouleRect.x = x + 100
    ampouleRect.y = y - 40
    if stat == 0:
        ecran.blit(ampoule[stat], ampouleRect)
        draw_path(ecran, x, y, dir)
    else:
        angle = 0
        ecran.blit(ampoule[stat], ampouleRect)
    rotate_image(ecran, x, y, img1, angle * dir)
    rotate_image(ecran, x + 100, y, img1, angle * dir)
    ecart = (angle / 360) * 50
    pygame.draw.line(ecran, BLACK, (x + ecart * (1), y - 15), (x
    + ecart * (1) + 10, y - 15), 3)
    pygame.draw.line(ecran, BLACK, (x + ecart * (1) + 50, y - 15)
    , (x + ecart * (1) + 10 + 50, y - 15), 3)
    pygame.draw.line(ecran, BLACK, (x + ecart * (-1) + 90, y +
    15), (x + ecart * (-1) + 100, y + 15), 3)
    pygame.draw.line(ecran, BLACK, (x + ecart * (-1) + 50, y +
    15), (x + ecart * (-1) + 10 + 50, y + 15), 3)

# Fonction pour dessiner des cercles animes
def draw_animated_circles(surface, x, y, radius, color,
    num_circles, step):
    for i in range(num_circles):
        angle = (pygame.time.get_ticks() / 100 + i * step) % 360
        offset_x = int(radius * cos(angle))
        offset_y = int(radius * sin(angle))
        pygame.draw.circle(surface, color, (x + offset_x, y +
    offset_y), 5)

# Fonction pour dessiner une barre de progression
def draw_progress_bar(surface, x, y, width, height, progress,
    color):
    pygame.draw.rect(surface, GREY, (x, y, width, height), 2)
    pygame.draw.rect(surface, color, (x, y, int(width * progress)
    , height))

# Fonction pour dessiner un texte centre
def draw_centered_text(surface, text, font, color, rect):
    text_surface = font.render(text, True, color)
    text_rect = text_surface.get_rect(center=rect.center)
    surface.blit(text_surface, text_rect)

```

```
# Fonction pour dessiner un tableau de bord
def draw_menu(surface, x, y, conv1, conv2, conv3, niv3, cycle):
    font = pygame.font.Font(None, 24)
    on_off = ["off", "on"]
    col = [RED, GREEN]
    # Arriere-plan du tableau de bord
    pygame.draw.rect(surface, WHiTE, (x, y, 300, 200))
    pygame.draw.rect(surface, BLACK, (x, y, 300, 200), 2)
    # Titres
    title_font = pygame.font.Font(None, 30)
    draw_centered_text(surface, "Tableau de Bord", title_font,
    BLACK, pygame.Rect(x, y, 300, 40))
    # Conveyeurs
    draw_centered_text(surface, f"Conv1: {on_off[conv1]}", font,
    col[conv1], pygame.Rect(x, y + 40, 300, 20))
    draw_centered_text(surface, f"Conv2: {on_off[conv2]}", font,
    col[conv2], pygame.Rect(x, y + 60, 300, 20))
    draw_centered_text(surface, f"Conv3: {on_off[conv3]}", font,
    col[conv3], pygame.Rect(x, y + 80, 300, 20))
    # Niveaux
    draw_centered_text(surface, f"Niveau 3: {niv3} Litre", font,
    BLACK, pygame.Rect(x, y + 100, 300, 20))
    draw_centered_text(surface, "Niveau 2: inf", font, BLACK,
    pygame.Rect(x, y + 120, 300, 20))
    draw_centered_text(surface, "Niveau 1: inf", font, BLACK,
    pygame.Rect(x, y + 140, 300, 20))
    # Cycle
    draw_centered_text(surface, f"Cycle: {cycle}", font, BLACK,
    pygame.Rect(x, y + 160, 300, 20))
    # Barre de progression pour le cycle
    draw_progress_bar(surface, x + 50, y + 190, 200, 10, cycle /
    100, BLUE)
    # Cercles animes
    draw_animated_circles(surface, x + 150, y + 100, 50, BLUE, 5,
     72)

def show_start_screen(ecran):
    # Couleurs et polices
    title_color = (255, 255, 255)
    button_color = (70, 130, 180)
    button_hover_color = (100, 149, 237)
    quit_button_color = (255, 0, 0)
    quit_button_hover_color = (220, 20, 60)
    title_font = pygame.font.Font(None, 74)
    button_font = pygame.font.Font(None, 50)

    # Texte du titre et des boutons
    title_text = title_font.render("Simulation de Reservoir",
    True, title_color)
    start_button_text = button_font.render("START", True,
    title_color)
    quit_button_text = button_font.render("QUIT", True,
    title_color)

    # Rectangles pour le titre et les boutons
    title_rect = title_text.get_rect(center=(400, 200))
    start_button_rect = pygame.Rect(275, 400, 250, 60)
```

```
    quit_button_rect = pygame.Rect(275, 500, 250, 60)

    # Arriere-plan anime
    background_surface = pygame.Surface((800, 650))
    for x in range(0, 800, 20):
        for y in range(0, 650, 20):
            pygame.draw.circle(background_surface, (40, 40, 40), (x, y), 2)

    waiting = True
    while waiting:
        ecran.fill(background_color)
        ecran.blit(background_surface, (0, 0))
        ecran.blit(title_text, title_rect)

        draw_tank(ecran, 350, 250, 100, 100, 100)
        # Bouton START
        mouse_pos = pygame.mouse.get_pos()
        if start_button_rect.collidepoint(mouse_pos):
            pygame.draw.rect(ecran, button_hover_color, start_button_rect)
        else:
            pygame.draw.rect(ecran, button_color, start_button_rect)
        ecran.blit(start_button_text, start_button_text.get_rect(center=start_button_rect.center))

        # Bouton QUiT
        if quit_button_rect.collidepoint(mouse_pos):
            pygame.draw.rect(ecran, quit_button_hover_color, quit_button_rect)
        else:
            pygame.draw.rect(ecran, quit_button_color, quit_button_rect)
        ecran.blit(quit_button_text, quit_button_text.get_rect(center=quit_button_rect.center))

        pygame.display.flip()

        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                pygame.quit()
                sys.exit()
            elif event.type == pygame.MOUSEBUTTONDOWN and event.button == 1:
                if start_button_rect.collidepoint(event.pos):
                    waiting = False
                elif quit_button_rect.collidepoint(event.pos):
                    pygame.quit()
                    sys.exit()



def main():
    continuer = True
    cycle = 0
    # Arriere-plan anime
```

```
243
244     # asset
245     roue = pygame.image.load('Assetes/roue.png')
246     angle = 0.0
247     vetesse_roue = 3.0
248
249     clock = pygame.time.Clock()
250
251     show_start_screen(ecran)
252     # Charger et jouer le son en boucle
253     pygame.mixer.music.load("sn.wav")
254     pygame.mixer.music.set_volume(0.3)
255     pygame.mixer.music.play(loops=-1)
256
257
258     while continuer:
259
260         for event in pygame.event.get():
261             if event.type == pygame.QUIT:
262                 continuer = False
263             elif event.type == pygame.KEYDOWN:
264                 if event.key == pygame.K_q:
265                     continuer = False
266                 elif event.key == pygame.K_UP:
267                     print("K_UP")
268                 elif event.key == pygame.K_DOWN:
269                     print("K_DOWN")
270         ecran.fill(GREY)  # Couleur de fond
271         fill_level = modbus.lireRegistre(30)
272         if fill_level == 10000:
273             cycle += 1
274         # conv
275         SUP_CONV1 = modbus.lireBit(303)
276         SUP_CONV2 = modbus.lireBit(304)
277         SUP_CONV3 = modbus.lireBit(305)
278         draw_menu(ecran, 5, 400, SUP_CONV1, SUP_CONV2, SUP_CONV3,
     fill_level, cycle)
279         draw_indicator(ecran, 200, 50, 100, 100, fill_level /
    100)
280
281         # Dessiner le reservoir avec le niveau de liquide
282         draw_tank(ecran, 200, 200, 100, 100, 100)
283         draw_tank(ecran, 500, 200, 100, 100, 100)
284         draw_tank(ecran, 350, 400, 100, 100, fill_level / 100)
285         draw_tank(ecran, 0, 600, 800, 50, 100)
286
287         angle += vetesse_roue
288         if angle >= 360:
289             angle = 0
290
291         if SUP_CONV3 == 1:
292             SUP_CONV3 = 0
293         else:
294             SUP_CONV3 = 1
295
296         if SUP_CONV2 == 1:
297             SUP_CONV2 = 0
```

```
298         else:
299             SUP_CONV2 = 1
300 
301         if SUP_CONV1 == 1:
302             SUP_CONV1 = 0
303         else:
304             SUP_CONV1 = 1
305 
306         tapie(ecran, 200, 350, angle, roue, 1, SUP_CONV1, AMPOULE
    )
307         tapie(ecran, 500, 350, angle, roue, -1, SUP_CONV2,
    AMPOULE)
308         tapie(ecran, 350, 550, angle, roue, 1, SUP_CONV3, AMPOULE
    )
309         clock.tick(60)
310 
311         pygame.display.flip()
312         clock.tick(30)
313 
314     pygame.quit()
315 
316 if __name__ == "__main__":
317     main()
```

Code source 1: Code source de la simulation de réservoir de liquide.